Information Builders®

WebFOCUS

Working With Cross-Century Dates

EDA, FIDEL, FOCUS, FOCUS Fusion, Information Builders, the Information Builders logo, TableTalk, and Web390 are registered trademarks and SiteAnalyzer, SmartMart, WebFOCUS, and WorldMART are trademarks of Information Builders, Inc.

Acrobat and Adobe are registered trademarks of Adobe Systems Incorporated.

AvantGo is a trademark of AvantGo, Inc.

WebLogic is a registered trademark of BEA Systems, Inc.

HP MPE/iX is a registered trademark of Hewlett Packard Corporation.

Informix is a registered trademark of Informix Software, Inc.

Intel is a registered trademark of Intel Corporation.

ACF/VTAM, AIX, AS/400, CICS, DB2, DRDA, Distributed Relational Database Architecture, IBM, MQSeries, MVS, OS/2, OS/390, OS/400, RACF, RS/6000, S/390, VM/ESA, and VTAM are registered trademarks and DB2/2, Hiperspace, IMS, MVS/ESA, QMF, SQL/DS, VM/XA and WebSphere are trademarks of International Business Machines Corporation.

INTERSOLVE and Q+E are registered trademarks of INTERSOLVE.

Orbix is a registered trademark of Iona Technologies Inc.

ActiveX, FrontPage, Microsoft, MS-DOS, PowerPoint, Visual Basic, Visual C++, Visual FoxPro, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Motorola is a trademark of Motorola, Inc.

Teradata is a registered trademark of NCR International, Inc.

Netscape, Netscape FastTrack Server, and Netscape Navigator are registered trademarks of Netscape Communications Corporation.

Nextel is a registered trademark of Nextel Communications.

NetWare and Novell are registered trademarks of Novell, Inc.

Oracle is a registered trademark and Rdb is a trademark of Oracle Corporation.

Palm is a trademark and Palm OS is a registered trademark of Palm Inc.

PeopleSoft is a registered trademark of PeopleSoft, Inc.

INFOAccess is a trademark of Pioneer Systems, Inc.

BlackBerry is a trademark and RIM is a registered trademark of Research In Motion Limited.

SAP and SAP R/3 are registered trademarks and SAP Business Information Warehouse and SAP BW are trademarks of SAP AG.

Java and all Java-based marks, NetDynamics, Solaris, SunOS, and iPlanet are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode is a trademark of Unicode, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2001, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

Contents

Working With Cross-Century Dates

Topics:

Many existing business applications use two digits to designate a year, instead of four digits. When they receive a value for a year, such as 00, they typically interpret it as 1900, assuming that the first two digits are 19, for the twentieth century. These applications require a way to handle dates when the century changes (for example, from the twentieth to the twenty-first), or when they need to perform comparisons or arithmetic on dates that span more than one century.

The cross-century date feature described in this topic enables the correct interpretation of the century if it is not explicitly provided, or is assumed to be the twentieth. The feature is application-based, that is, it involves modifications to procedures or metadata so that dates are accurately interpreted and processed. The feature is called the sliding window technique.

When Do You Use the Sliding Window Technique?

If your application accesses dates that contain an explicit century, the century is accepted as is. Your application can run correctly across centuries, and you do not need to use the sliding window technique.

If your application accesses dates without explicit centuries, they assume the default value 19. Your application will require remediation, such as the sliding window technique, to ensure the correct interpretation of the century if the default is not valid, and to run as expected in the next century.

This topic does not cover remediation options such as date expansion, which requires that data be changed in the data source to accommodate explicit century values. For a list of Information Builders documentation on remediation, see your latest *Publications Catalog*.

This topic covers the use of the sliding window technique in reporting applications. Details on when to use the sliding window technique are provided later in this topic. It also includes reference information on the use of the technique with FOCUS MODIFY requests. For additional information on implementing this technique with Maintain, see your database maintenance documentation. References to MODIFY and Maintain apply only to Developer Studio.

The Sliding Window Technique

With the sliding window technique, you do not need to change stored data from a 2-digit year format to a 4-digit year format in order to determine the century. Instead, you can continue storing 2-digit years and expand them when your application accesses them.

The sliding window technique recognizes that the earliest and latest values for a single date field in most business applications are within 100 years of one another. For example, a human resources application typically contains a field for the birth date of each active employee. The difference in the birth date (or age) of the oldest active employee and the youngest active employee is not likely to be more than 100.

The technique is implemented as follows:

- You define the start of a 100-year sliding window by supplying two values: one for the default century (DEFCENT) and one for the year threshold (YRTHRESH). For example, a value of 19 for the century, combined with a value of 60 for the threshold, creates a window that starts in 1960 and ends in 2059.
- The threshold provides a way to assign a value to the century of a 2-digit year:
    - A year greater than or equal to the threshold assumes the value of the default century (DEFCENT). Using the sample value 19 for the default century and 60 for the threshold, a 2-digit year of 70 is interpreted as 1970 (70 is greater than 60).

- A year less than the threshold assumes the value of the default century plus 1 (DEFCENT + 1). Using the same sample values (19 and 60), a 2-digit year of 50 is interpreted as 2050 (50 is less than 60), and a 2-digit year of 00 is interpreted as 2000 (00 is also less than 60).

The conversion rule for this example is illustrated as follows:

```
0 ______________________ < YRTHRESH = 60 ≥ ____________________ 99
            ⇑                                      ⇑
  Century = DEFCENT + 1 (20)           Century = DEFCENT (19)
```

Any 2-digit year is assumed to fall within the window. You must handle dates that fall outside the defined window by coding.

Each file or each date field used in an application can have its own conversion rule, which provides the flexibility required by most applications.

Defining a Sliding Window

You can define a sliding window in several ways, depending on the specific requirements of your application:

- **Globally.** The SET DEFCENT and SET YRTHRESH commands define a window on a global level.
- **On a file level.** The FDEFCENT and FYRTHRESH attributes in a Master File define a window on a file level, allowing the correct interpretation of date fields from multiple files that span different time periods.
- **On a field level.** The DEFCENT and YRTHRESH attributes in a Master File define a window on a field level, allowing the correct interpretation of date fields, within a single file, that span different time periods.
- **For a virtual field.** The DEFCENT and YRTHRESH parameters on a DEFINE command, in either a request or a Master File, define a window for a virtual field.
- **For a calculated value.** The DEFCENT and YRTHRESH parameters on a COMPUTE command define a window for a calculated value.

If you define more than one window using any of the preceding methods, the precedence is as follows:

1. DEFCENT and YRTHRESH on a DEFINE or COMPUTE command.
2. DEFCENT and YRTHRESH field-level attributes in a Master File.
3. FDEFCENT and FYRTHRESH file-level attributes in a Master File.
4. SET DEFCENT and SET YRTHRESH on a global level; if you do not specify values, the defaults are used (DEFCENT = 19, YRTHRESH = 0).

Creating a Dynamic Window Based on the Current Year

An optional feature of the sliding window technique enables you to create a dynamic window, defining the start of a 100-year span based on the current year. The start year and threshold for the window automatically change at the beginning of each new year.

If an application requires that a window's start year change when a new year begins, use of this feature avoids the necessity of manually re-coding it.

To implement this feature, YRTHRESH or FYRTHRESH is offset from the current year, or given a negative value.

For example, if the current year is 1999 and YRTHRESH is set to -38, a window from 1961 to 2060 is created. The start year 1961 is derived by subtracting 38 (the value of YRTHRESH) from 1999 (the current year). To interpret dates that fall within this window, the threshold 61 is used.

At the beginning of the year 2000, a new window from 1962 to 2061 is automatically created; for dates that fall within this window, the threshold 62 is used. In the year 2001, the window becomes 1963 to 2062, and the threshold is 63, and so on.

With each new year, the start year for the window is incremented by one.

When using this feature, do not code a value for DEFCENT or FDEFCENT, since the feature is designed to automatically calculate the value for the default century. Be aware of the following:

- If you do code a value for DEFCENT on the field level in a Master File, or for FDEFCENT on the file level in a Master File, the feature will not work as intended. The value for the century, which is automatically calculated by YRTHRESH by design, will be reset to the value you code for DEFCENT or FDEFCENT.
- If you code a value for DEFCENT anywhere other than the field level in a Master File (for example, on the global level), and YRTHRESH is negative, the coded value will be ignored. The default century will be automatically calculated as designed.

Applying the Sliding Window Technique

To apply the sliding window technique correctly, you need to understand the difference between a date format (formerly called a smart date) and a legacy date:

- A date format refers to an internally stored integer that represents the number of days between a real date value and a base date (either December 31, 1900, for dates with YMD or YYMD format; or January 1901, for dates with YM, YYM, YQ, or YYQ format). A Master File does not specify a data type or length for a date format; instead, it specifies display options such as D (day), M (month), Y (2-digit year), or YY (4-digit year). For example, MDYY in the USAGE (also known as FORMAT) attribute of a Master File is a date format. A real date value such as March 5, 1999, displays as 03/05/1999, and is internally stored as the offset from December 31, 1900.
- A legacy date refers to an integer, packed decimal, double precision, floating point, or alphanumeric format with date edit options, such as I6YMD, A6MDY, I8YYMD, or A8MDYY. For example, A6MDY is a 6-byte alphanumeric string; the suffix MDY indicates how Information Builders will return the data in the field. The sample value 030599 displays as 03/05/99.

For details on date fields, see your documentation on describing data.

When to Supply Settings for DEFCENT and YRTHRESH

The rest of this topic refers simply to DEFCENT when either DEFCENT or FDEFCENT applies, and to YRTHRESH when either YRTHRESH or FYRTHRESH applies.

Supply settings for DEFCENT and YRTHRESH in the following cases:

- When you issue a DEFINE or COMPUTE command to convert a legacy date without century digits to a date format with century digits (for example, to convert the format I6YMD to YYMD). With DEFINE and COMPUTE, DEFCENT and YRTHRESH do not work directly on legacy dates; for example, you cannot use them to convert the legacy date format I6YMD to the legacy date format I8YYMD.
- When a DEFINE command, COMPUTE command, or Dialogue Manager -SET command calls a function, supplied by Information Builders, that uses legacy dates, and the input date does not contain century digits.

  On input, the function will use the window defined for an I6 legacy date field (with edit options). The output format may be I8 (again, with edit options), which includes a 4-digit year.
- When data is entered or changed in a date format field in a FOCUS data source, or an SQL date is entered or changed in a Relational Database Management System (RDBMS), and the input date does not contain century digits.

For example, in Developer Studio you can use the sliding window technique in applications that use FIXFORM or CRTFORM with MODIFY.

- When a data source is read, and the ACTUAL attribute in the Master File is non-date specific (for example, A6, I6, or P6), without century digits, and the FORMAT or USAGE attribute specifies a date format. This case does not apply to FOCUS data sources.

Follow these rules when implementing the sliding window technique:

- Specify values for both DEFCENT and YRTHRESH to ensure consistent coding and accurate results, except when YRTHRESH has a negative value. In that case, specify a value for YRTHRESH only; do not code a value for DEFCENT.
- Do not use DEFCENT and YRTHRESH with ON TABLE SET.

Finally, keep in mind that the sliding window technique does not change the way existing data is stored. Rather, it accurately interprets data during application processing.

Reference Restrictions With MODIFY

Note: This topic applies to Developer Studio only.

The following results occur when you use the sliding window technique with a MODIFY request or FOCCOMP procedure:

- A MODIFY request compiled prior to Version 7.0 Release 6, when run with global SET DEFCENT and SET YRTHRESH settings, or with file-level or field-level settings, yields a FOC1886 error message. You must recompile the MODIFY request.
- A MODIFY request compiled in Version 7.0 Release 6, when run with global SET DEFCENT and SET YRTHRESH settings, or with file-level or field-level settings, yields a FOC1885 warning message.
- A FOCCOMP procedure, compiled with global SET DEFCENT and SET YRTHRESH settings, and run in releases prior to Version 7.0 Release 6, yields a FOC548 invalid version message. You must recompile the MODIFY request.
- A FOCCOMP procedure that contains DEFCENT/YRTHRESH or FDEFCENT/FYRTHRESH attributes in the associated Master File, and run in releases prior to Version 7.0 Release 6, yields a FOC306 description error message.

Date Validation

Date formats are validated on input. For example, 11/99/1999 is rejected as input to a date field formatted as MDYY, because 99 is not a valid day. Information Builders generates an error message.

Legacy dates are not validated. The date 11991999, described with the format A8MDYY, is accepted, even though it, too, contains the invalid day 99.

Defining a Global Window With SET

The SET DEFCENT and SET YRTHRESH commands define a window on a global level. The time span created by the SET commands applies to every 2-digit year used by the application unless you specify file-level or field-level windows elsewhere.

For details on specifying parameters that govern the environment, see your documentation on the SET command.

Syntax

How to Define a Global Window With SET

To define a global window, issue two SET commands.

The first command is

```
SET DEFCENT = {cc|19}
```

where:

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

The second command is

```
SET YRTHRESH = {[-]yy|0}
```

where:

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of FDEFCENT for the century. Two-digit years less than the threshold assume the value of FDEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and FDEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example

Defining a Global Window With SET

In the following request, the SET command defines a global window from 1983 to 2082.

As SET syntax allows, the command is entered on one line, with the parameters separated by a comma. You do not need to repeat the keyword SET for YRTHRESH.

The DEFINE command converts the legacy date EFFECT_DATE into the date format NEW_DATE. It creates NEW_DATE as a virtual field, derived from the existing field EFFECT_DATE. The format of EFFECT_DATE is I6YMD, which is a 2-digit year. NEW_DATE is formatted as YYMD, which is a 4-digit year. For details on DEFINE, see your documentation on creating reports.

The request is:

```
SET DEFCENT = 19, YRTHRESH = 83

DEFINE FILE EMPLOYEE
NEW_DATE/YYMD = EFFECT_DATE;
END

TABLE FILE EMPLOYEE
PRINT EFFECT_DATE NEW_DATE BY EMP_ID
END
```

In the report, the value of the 2-digit year 82 is less than the threshold 83, so it assumes the value 20 for the century (DEFCENT + 1) and is returned as 2082 in the NEW_DATE column. The other year values (83 and 84) are greater than or equal to the threshold 83, so their century defaults to the value 19 (DEFCENT); they are returned as 1983 and 1984 under NEW_DATE.

The output is:

```
EMP_ID      EFFECT_DATE    NEW_DATE
071382660
112847612
117593129   82/11/01       2082/11/01
119265415
119329144   83/01/01       1983/01/01
123764317   83/03/01       1983/03/01
126724188
219984371
326179357   82/12/01       2082/12/01
451123478   84/09/01       1984/09/01
543729165
818692173   83/05/01       1983/05/01
```

In the example, missing date values appear as blanks by default. To retrieve the base date value for the NEW_DATE field instead of blanks, issue the command

```
SET DATEDISPLAY = ON
```

before running the request.

The base date value for NEW_DATE, which is formatted as YYMD, is returned as 1900/12/31:

```
EMP_ID      EFFECT_DATE      NEW_DATE
071382660                    1900/12/31
112847612                    1900/12/31
117593129   82/11/01         2082/11/01
119265415                    1900/12/31
119329144   83/01/01         1983/01/01
123764317   83/03/01         1983/03/01
126724188                    1900/12/31
219984371                    1900/12/31
326179357   82/12/01         2082/12/01
451123478   84/09/01         1984/09/01
543729165                    1900/12/31
818692173   83/05/01         1983/05/01
```

If NEW_DATE had a YYM format, the base date would appear as 1901/01. If it had a YYQ format, it would appear as 1901 Q1.

If the value of NEW_DATE is 0 and SET DATEDISPLAY = OFF (the default), blanks are displayed. With SET DATEDISPLAY = ON, the base date is displayed instead of blanks. Zero (0) is treated as an offset from the base date, which results in the base date.

For details on SET DATEDISPLAY, see your documentation on the SET command.

Defining a Dynamic Global Window With SET

This topic illustrates the creation of a dynamic window using the global command SET YRTHRESH. You can also implement this feature on the file and field level, and on a DEFINE or COMPUTE.

With this option of the sliding window technique, the start year and threshold for the window automatically change at the beginning of each new year. The default century (DEFCENT) is automatically calculated.

You can use SET TESTDATE to alter the system date when testing a dynamic window (that is, when YRTHRESH has a negative value). However, when testing a dynamic window defined in a Master File, you must issue a CHECK FILE command each time you issue a SET TESTDATE command. CHECK FILE reloads the Master File into memory and ensures the correct recalculation of the start date of the dynamic window. For details on SET TESTDATE, see your documentation on the SET command. For details on CHECK FILE, see your documentation on describing data.

Example **Defining a Dynamic Global Window With SET**

In the following request, the COMPUTE command calls the function AYMD, supplied by Information Builders. AYMD adds one day to the input field, HIRE_DATE; the output field, HIRE_DATE_PLUS_ONE, contains the result. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. HIRE_DATE_PLUS_ONE is formatted as I8YYMD, which is a legacy date with a 4-digit year.

The function uses the YRTHRESH value set at the beginning of the request to create a dynamic window for the input field HIRE_DATE. The start date of the window is incremented by one at the beginning of each new year. Notice that DEFCENT is not coded, since the default century is automatically calculated whenever YRTHRESH has a negative value.

The function inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

Sample values are shown in the reports for 1999, 2000, and 2018, which follow the request.

For details on AYMD, see your documentation on creating reports.

The request is:

```
SET YRTHRESH = -18

TABLE FILE EMPLOYEE
PRINT HIRE_DATE AND COMPUTE
     HIRE_DATE_PLUS_ONE/I8YYMD = AYMD(HIRE_DATE, 1, HIRE_DATE_PLUS_ONE);
END
```

In 1999, the window spans the years 1981 to 2080. The threshold is 81 (1999 - 18). In the report, the 2-digit year 80 is less than the threshold 81, so it assumes the value 20 for the century (DEFCENT + 1), and is returned as 2080 in the HIRE_DATE_PLUS_ONE column. The other year values (81 and 82) are greater than or equal to the threshold 81, so their century defaults to the value of DEFCENT (19); they are returned as 1981 and 1982.

The output is:

```
HIRE_DATE     HIRE_DATE_PLUS_ONE
80/06/02      2080/06/03
81/07/01      1981/07/02
82/05/01      1982/05/02
82/01/04      1982/01/05
82/08/01      1982/08/02
82/01/04      1982/01/05
82/07/01      1982/07/02
81/07/01      1981/07/02
82/04/01      1982/04/02
82/02/02      1982/02/03
82/04/01      1982/04/02
81/11/02      1981/11/03
```

In 2000, the window spans the years 1982 to 2081. The threshold is 82 (2000 - 18). In the report, the 2-digit years 80 and 81 are less than the threshold; for the century, they assume the value 20 (DEFCENT + 1). The 2-digit year 82 is equal to the threshold; for the century, it defaults to the value 19 (DEFCENT).

The output is:

```
HIRE_DATE     HIRE_DATE_PLUS_ONE
80/06/02      2080/06/03
81/07/01      2081/07/02
82/05/01      1982/05/02
82/01/04      1982/01/05
82/08/01      1982/08/02
82/01/04      1982/01/05
82/07/01      1982/07/02
81/07/01      2081/07/02
82/04/01      1982/04/02
82/02/02      1982/02/03
82/04/01      1982/04/02
81/11/02      2081/11/03
```

Running the report in 2018 illustrates the automatic recalculation of DEFCENT from 19 to 20. In 2018, the window spans the years 2000 to 2099. The threshold is 0 (2018 - 18). A 2-digit year greater than or equal to 0 defaults to the recalculated value 20 (DEFCENT).

Since all the values for the HIRE_DATE year are greater than 0, their century defaults to 20.

The output is:

```
HIRE_DATE    HIRE_DATE_PLUS_ONE
80/06/02     2080/06/03
81/07/01     2081/07/02
82/05/01     2082/05/02
82/01/04     2082/01/05
82/08/01     2082/08/02
82/01/04     2082/01/05
82/07/01     2082/07/02
81/07/01     2081/07/02
82/04/01     2082/04/02
82/02/02     2082/02/03
82/04/01     2082/04/02
81/11/02     2081/11/03
```

Querying the Current Global Value of DEFCENT and YRTHRESH

You can query the current global value of DEFCENT and YRTHRESH.

Syntax **How to Query the Current Global Value of DEFCENT and YRTHRESH**

```
? SET DEFCENT
? SET YRTHRESH
```

where:

`DEFCENT`

Returns the value for the DEFCENT parameter.

`YRTHRESH`

Returns the value for the YRTHRESH parameter.

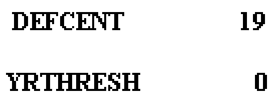
Example **Querying the Current Global Value of DEFCENT and YRTHRESH**

Enter

```
? SET DEFCENT
? SET YRTHRESH
```

to query the current global value of DEFCENT and YRTHRESH.

The following is a response to the query:



```
 DEFCENT          19

YRTHRESH           0
```

Defining a File-Level or Field-Level Window in a Master File

In this implementation of the sliding window technique, you change the metadata used by an application. Two pairs of Master File attributes enable you to define a window on a file or field level:

- The FDEFCENT and FYRTHRESH attributes define a window on a file level. They enable the correct interpretation of legacy date fields from multiple files that span different time periods.

  A file-level window takes precedence over a global window for the dates associated with that file.

- The DEFCENT and YRTHRESH attributes define a window on a field level, enabling the correct interpretation of legacy date fields, within a single file, that span different time periods. Each legacy date field in a file can have its own window. For example, in an insurance application, the range of dates for date of birth may be from 1910 to 2009, and the range of dates for expected death may be from 1990 to 2089.

  A field-level window takes precedence over a file-level or global window for the dates associated with that field.

For details on Master Files, see your documentation on describing data.

Syntax How to Define a File-Level Window in a Master File

To define a window that applies to all legacy date fields in a file, add the FDEFCENT and FYRTHRESH attributes to the Master File on the file declaration.

The syntax for the first attribute is

```
{FDEFCENT|FDFC} = {cc|19}
```

where:

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

The syntax for the second attribute is

```
{FYRTHRESH|FYRT} = {[-]yy|0}
```

where:

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example Defining a File-Level Window in a Master File

Tip: Use the abbreviated forms of FDEFCENT/FYRTHRESH or DEFCENT/YRTHRESH to reduce keystrokes. The examples in this topic use the abbreviated forms where available (for instance, FDFC instead of FDEFCENT). Maintain supports only the abbreviated forms in certain command syntax (for example, on a COMPUTE or DECLARE command). For details, see your database maintenance documentation.

In the following example, the FDEFCENT and FYRTHRESH attributes define a window from 1982 to 2081. The window is applied to all legacy date fields in the file, including HIRE_DATE, DAT_INC, and others, if they are converted to a date format.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=82
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,       ALIAS=EID,    FORMAT=A9,      $
  FIELDNAME=LAST_NAME,    ALIAS=LN,     FORMAT=A15,     $
  FIELDNAME=FIRST_NAME,   ALIAS=FN,     FORMAT=A10,     $
  FIELDNAME=HIRE_DATE,    ALIAS=HDT,    FORMAT=I6YMD,   $
.
.
.
  FIELDNAME=DAT_INC,      ALIAS=DI,     FORMAT=I6YMD,   $
.
.
.
```

The DEFINE command in the following request creates two virtual fields named NEW_HIRE_DATE, which is derived from the existing field HIRE_DATE; and NEW_DAT_INC, which is derived from DAT_INC. The format of HIRE_DATE and DAT_INC is I6YMD, which is a legacy date with a 2-digit year. NEW_HIRE_DATE and NEW_DAT_INC are date formats with 4-digit years (YYMD). For details on DEFINE, see your documentation on creating reports.

```
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE DAT_INC NEW_DAT_INC
END
```

The window created in the Master File applies to both legacy date fields. In the report, the year 82 (which is equal to the threshold), for both HIRE_DATE and DAT_INC, defaults to the century value 19 and is returned as 1982 in the NEW_HIRE_DATE and NEW_DAT_INC columns. The year 81, for both HIRE_DATE and DAT_INC, is less than the threshold 82 and assumes the century value 20 (FDEFCENT + 1).

The output is:

```
HIRE_DATE    NEW_HIRE_DATE    DAT_INC    NEW_DAT_INC
80/06/02     2080/06/02       82/01/01   1982/01/01
80/06/02     2080/06/02       81/01/01   2081/01/01
81/07/01     2081/07/01       82/01/01   1982/01/01
82/05/01     1982/05/01       82/06/01   1982/06/01
82/05/01     1982/05/01       82/05/01   1982/05/01
82/01/04     1982/01/04       82/05/14   1982/05/14
82/01/04     1982/01/04       82/01/04   1982/01/04
82/08/01     1982/08/01       82/08/01   1982/08/01
82/01/04     1982/01/04       82/05/14   1982/05/14
82/01/04     1982/01/04       82/01/04   1982/01/04
82/07/01     1982/07/01       82/07/01   1982/07/01
81/07/01     2081/07/01       82/01/01   1982/01/01
82/04/01     1982/04/01       82/04/01   1982/04/01
82/02/02     1982/02/02       82/05/14   1982/05/14
82/02/02     1982/02/02       82/02/02   1982/02/02
82/04/01     1982/04/01       82/06/11   1982/06/11
82/04/01     1982/04/01       82/04/01   1982/04/01
81/11/02     2081/11/02       82/04/09   1982/04/09
81/11/02     2081/11/02       81/11/02   2081/11/02
```

Syntax How to Define a Field-Level Window in a Master File

To define a window that applies to a specific legacy date field, add the DEFCENT and YRTHRESH attributes to the Master File on the field declaration.

The syntax for the first attribute is

```
{DEFCENT|DFC} = {cc|19}
```

where:

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

The syntax for the second attribute is

```
{YRTHRESH|YRT} = {[-]yy|0}
```

where:

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example Defining a Field-Level Window in a Master File

In this example, the application requires a different window for two legacy date fields in the same file.

The DEFCENT and YRTHRESH attributes in the Master File define a window for HIRE_DATE from 1982 to 2081, and a window for DAT_INC from 1983 to 2082.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,        ALIAS=EID,    FORMAT=A9,                        $
  FIELDNAME=LAST_NAME,     ALIAS=LN,     FORMAT=A15,                       $
  FIELDNAME=FIRST_NAME,    ALIAS=FN,     FORMAT=A10,                       $
  FIELDNAME=HIRE_DATE,     ALIAS=HDT,    FORMAT=I6YMD, DFC=19, YRT=82,     $
.
.
.
  FIELDNAME=DAT_INC,       ALIAS=DI,     FORMAT=I6YMD, DFC=19, YRT=83,     $
.
.
.
```

The request is the same one used in the previous example (defining a file-level window in a Master File):

```
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE DAT_INC NEW_DAT_INC
END
```

However, the report illustrates the use of two different windows for the two legacy date fields. For example, the year 82 for HIRE_DATE defaults to the century value 19, since 82 is equal to the threshold for the window for this field. The date returned for NEW_HIRE_DATE is 1982.

The year 82 for DAT_INC assumes the century value 20 (DEFCENT + 1), since 82 is less than the threshold for the window for this field (83). The date returned for NEW_DAT_INC is 2082.

The output is:

```
HIRE_DATE   NEW_HIRE_DATE   DAT_INC   NEW_DAT_INC
80/06/02    2080/06/02      82/01/01  2082/01/01
80/06/02    2080/06/02      81/01/01  2081/01/01
81/07/01    2081/07/01      82/01/01  2082/01/01
82/05/01    1982/05/01      82/06/01  2082/06/01
82/05/01    1982/05/01      82/05/01  2082/05/01
82/01/04    1982/01/04      82/05/14  2082/05/14
82/01/04    1982/01/04      82/01/04  2082/01/04
82/08/01    1982/08/01      82/08/01  2082/08/01
82/01/04    1982/01/04      82/05/14  2082/05/14
82/01/04    1982/01/04      82/01/04  2082/01/04
82/07/01    1982/07/01      82/07/01  2082/07/01
81/07/01    2081/07/01      82/01/01  2082/01/01
82/04/01    1982/04/01      82/04/01  2082/04/01
82/02/02    1982/02/02      82/05/14  2082/05/14
82/02/02    1982/02/02      82/02/02  2082/02/02
82/04/01    1982/04/01      82/06/11  2082/06/11
82/04/01    1982/04/01      82/04/01  2082/04/01
81/11/02    2081/11/02      82/04/09  2082/04/09
81/11/02    2081/11/02      81/11/02  2081/11/02
```

Example Defining a Field-Level Window in a Master File Used With MODIFY

This example illustrates the use of field-level DEFCENT and YRTHRESH attributes to define a window used with MODIFY. To run this example yourself, you need to create a Master File named DATE and a procedure named DATELOAD.

The Master File describes a segment with 12 date fields of different formats. The first field is a date format field. The DEFCENT and YRTHRESH attributes included on this field create a window from 1990 to 2089. The window is required because the input data for the first date field does not contain century digits, and the default value 19 cannot be assumed.

The Master File looks like this:

```
FILENAME=DATE, SUFFIX=FOC
 SEGNAME=ONE,  SEGTYPE=S1
  FIELDNAME=D1_YYMD,   ALIAS=D1,   FORMAT=YYMD, DFC=19, YRT=90,  $
  FIELDNAME=D2_I6YMD,  ALIAS=D2,   FORMAT=I6YMD,                 $
  FIELDNAME=D3_I8YYMD, ALIAS=D3,   FORMAT=I8,                    $
  FIELDNAME=D4_A6YMD,  ALIAS=D4,   FORMAT=A6YMD,                 $
  FIELDNAME=D5_A8YYMD, ALIAS=D5,   FORMAT=A8YYMD,                $
  FIELDNAME=D6_I4YM,   ALIAS=D6,   FORMAT=I4YM,                  $
  FIELDNAME=D7_YQ,     ALIAS=D7,   FORMAT=YQ,                    $
  FIELDNAME=D8_YM,     ALIAS=D8,   FORMAT=YM,                    $
  FIELDNAME=D9_JUL,    ALIAS=D9,   FORMAT=JUL,                   $
  FIELDNAME=D10_Y,     ALIAS=D10,  FORMAT=Y,                     $
  FIELDNAME=D11_YY,    ALIAS=D11,   FORMAT=YY,                   $
  FIELDNAME=D12_MDYY,  ALIAS=D12,  FORMAT=MDYY,                  $
```

The procedure (DATELOAD) creates a FOCUS data source named DATE and loads two records into it. The first field of the first record contains the 2-digit year 92. The first field of the second record contains the 2-digit year 88. For details on commands such as CREATE and MODIFY, and others used in this file, see your database maintenance documentation.

The procedure looks like this:

```
CREATE FILE DATE
MODIFY FILE DATE
FIXFORM D1/8 D2/6 D3/8 D4/6 D5/8 D6/4 D7/4 D8/4 D9/5 D10/2 D11/4 D12/8
MATCH D1
   ON NOMATCH INCLUDE
   ON MATCH REJECT
DATA
  920229000229200002290002292000022900020001000200060002000022920000
  880229000229200002290002292000022900020001000200060002000022920000
END
```

The following request accesses all the fields in the new data source:

```
TABLE FILE DATE
PRINT *
END
```

In the report, the year 92 for D1_YYMD defaults to the century value 19, since 92 is greater than the threshold for the window for this field (90). It is returned as 1992 in the D1_YYMD column. The year 88 assumes the century value 20 (DEFCENT + 1), because 88 is less than the threshold. It is returned as 2088 in the D1_YYMD column.

The partial output is:

```
D1_YYMD      D2_I6YMD     D3_I8YYMD     D4_A6YMD     D5_A8YYMD     D6_I4YM    D7_YQ

1992/02/29   00/02/29     20000229      00/02/29     2000/02/29    00/02      00 Q1

2088/02/29   00/02/29     20000229      00/02/29     2000/02/29    00/02      00 Q1
```

Example **Defining Both File-Level and Field-Level Windows**

The following Master File defines windows at both the file and field level:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=83
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,        ALIAS=EID,     FORMAT=A9,                  $
  FIELDNAME=LAST_NAME,     ALIAS=LN,      FORMAT=A15,                 $
  FIELDNAME=FIRST_NAME,    ALIAS=FN,      FORMAT=A10,                 $
  FIELDNAME=HIRE_DATE,     ALIAS=HDT,     FORMAT=I6YMD, DFC=19, YRT=82, $
.
.
.
  FIELDNAME=EFFECT_DATE,   ALIAS=EDATE,   FORMAT=I6YMD,               $
.
.
.

  FIELDNAME=DAT_INC,       ALIAS=DI,      FORMAT=I6YMD,               $
.
.
.
```

The request is:

```
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_EFFECT_DATE/YYMD = EFFECT_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE EFFECT_DATE NEW_EFFECT_DATE DAT_INC
NEW_DAT_INC
END
```

When the field HIRE_DATE is accessed, the time span 1982 to 2081 is applied. For all other legacy date fields in the file, such as EFFECT_DATE and DAT_INC, the time span specified at the file level is applied, that is, 1983 to 2082.

For example, the year 82 for HIRE_DATE is returned as 1982 in the NEW_HIRE_DATE column, since 82 is equal to the threshold of the window for that particular field. The year 82 for EFFECT_DATE and DAT_INC is returned as 2082 in the columns NEW_EFFECT_DATE and NEW_DAT_INC, since 82 is less than the threshold of the file-level window (83).

The output is:

```
HIRE_DATE  NEW_HIRE_DATE  EFFECT_DATE  NEW_EFFECT_DATE  DAT_INC   NEW_DAT_INC
80/06/02   2080/06/02                                   82/01/01  2082/01/01
80/06/02   2080/06/02                                   81/01/01  2081/01/01
81/07/01   2081/07/01                                   82/01/01  2082/01/01
82/05/01   1982/05/01     82/11/01     2082/11/01       82/06/01  2082/06/01
82/05/01   1982/05/01     82/11/01     2082/11/01       82/05/01  2082/05/01
82/01/04   1982/01/04                                   82/05/14  2082/05/14
82/01/04   1982/01/04                                   82/01/04  2082/01/04
82/08/01   1982/08/01     83/01/01     1983/01/01       82/08/01  2082/08/01
82/01/04   1982/01/04     83/03/01     1983/03/01       82/05/14  2082/05/14
82/01/04   1982/01/04     83/03/01     1983/03/01       82/01/04  2082/01/04
82/07/01   1982/07/01                                   82/07/01  2082/07/01
81/07/01   2081/07/01                                   82/01/01  2082/01/01
82/04/01   1982/04/01     82/12/01     2082/12/01       82/04/01  2082/04/01
82/02/02   1982/02/02     84/09/01     1984/09/01       82/05/14  2082/05/14
82/02/02   1982/02/02     84/09/01     1984/09/01       82/02/02  2082/02/02
82/04/01   1982/04/01                                   82/06/11  2082/06/11
82/04/01   1982/04/01                                   82/04/01  2082/04/01
81/11/02   2081/11/02     83/05/01     1983/05/01       82/04/09  2082/04/09
81/11/02   2081/11/02     83/05/01     1983/05/01       81/11/02  2081/11/02
```

Missing date values for NEW_EFFECT_DATE appear as blanks by default. To retrieve the base date value for NEW_EFFECT_DATE instead of blanks, issue the command

```
SET DATEDISPLAY = ON
```

before running the request. The base date value is returned as 1900/12/31. See *Defining a Global Window With SET* on page 7 for sample results.

Defining a Window for a Virtual Field

The DEFCENT and YRTHRESH parameters on a DEFINE command create a window for a virtual field. The window is used to interpret date values for the virtual field when the century is not supplied. You can issue a DEFINE command in either a request or a Master File.

The DEFCENT and YRTHRESH parameters must immediately follow the field format specification; their values are always taken from the left side of the DEFINE syntax (that is, from the left side of the equal sign). If the expression in the DEFINE contains a function call, the function uses the DEFCENT and YRTHRESH values for the input field. The standard order of precedence (field level/file level/global level) applies to the DEFCENT and YRTHRESH values for the input field.

Syntax How to Define a Window for a Virtual Field in a Request

Use standard DEFINE syntax for a request, as described in your documentation on creating reports. Partial DEFINE syntax is shown here.

On the line that specifies the name of the virtual field, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
DEFINE FILE filename
  fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =
    expression;
.
.
.
END
```

where:

filename

Is the name of the file for which you are creating the virtual field.

fieldname

Is the name of the virtual field.

format

Is a date format such as DMY or YYMD.

DEFCENT

Is the parameter for the default century.

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

YRTHRESH

Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

`expression`

Is a valid arithmetic or logical expression, function, or function that determines the value of the virtual field.

`END`

Is required to terminate the DEFINE command.

Example Defining a Window for a Virtual Field in a Request

In the following request, the DEFINE command creates two virtual fields, GLOBAL_HIRE_DATE and WINDOWED_HIRE_DATE. Both virtual fields are derived from the existing field HIRE_DATE. The format of HIRE_DATE is I6YMD, which is a legacy date with a 2-digit year. The virtual fields are date formats with a 4-digit year (YYMD).

The second virtual field, WINDOWED_HIRE_DATE, has the additional parameters DEFCENT and YRTHRESH, which define a window from 1982 to 2081. Notice that both DEFCENT and YRTHRESH are coded, as required.

The request is:

```
DEFINE FILE EMPLOYEE
GLOBAL_HIRE_DATE/YYMD = HIRE_DATE;
WINDOWED_HIRE_DATE/YYMD DFC 19 YRT 82 = HIRE_DATE;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE GLOBAL_HIRE_DATE WINDOWED_HIRE_DATE
END
```

Assuming that there are no FDEFCENT and FYRTHRESH file-level settings in the Master File for EMPLOYEE, the global default settings (DEFCENT = 19, YRTHRESH = 0) are used to interpret 2-digit years for HIRE_DATE when deriving the value of GLOBAL_HIRE_DATE. For example, the value of all years for HIRE_DATE (80, 81, and 82) is greater than 0; consequently they default to 19 for the century and are returned as 1980, 1981, and 1982 in the GLOBAL_HIRE_DATE column.

For WINDOWED_HIRE_DATE, the window created specifically for that field (1982 to 2081) is used. The 2-digit years 80 and 81 for HIRE_DATE are less than the threshold for the window (82); consequently, they are returned as 2080 and 2081 in the WINDOWED_HIRE_DATE column.

The output is:

```
HIRE_DATE    GLOBAL_HIRE_DATE    WINDOWED_HIRE_DATE
80/06/02     1980/06/02          2080/06/02
81/07/01     1981/07/01          2081/07/01
82/05/01     1982/05/01          1982/05/01
82/01/04     1982/01/04          1982/01/04
82/08/01     1982/08/01          1982/08/01
```

```
82/01/04      1982/01/04          1982/01/04
82/07/01      1982/07/01          1982/07/01
81/07/01      1981/07/01          2081/07/01
82/04/01      1982/04/01          1982/04/01
82/02/02      1982/02/02          1982/02/02
82/04/01      1982/04/01          1982/04/01
81/11/02      1981/11/02          2081/11/02
```

Example Defining a Window for Function Input in a DEFINE Command

The following sample request illustrates a call to the function AYMD in a DEFINE command. AYMD adds 60 days to the input field, HIRE_DATE; the output field, SIXTY_DAYS, contains the result. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. SIXTY_DAYS is formatted as I8YYMD, which is a legacy date with a 4-digit year.

For details on AYMD, see your documentation on creating reports.

```
DEFINE FILE EMPLOYEE
SIXTY_DAYS/I8YYMD = AYMD(HIRE_DATE, 60, 'I8YYMD');
END
```

```
TABLE FILE EMPLOYEE
PRINT HIRE_DATE SIXTY_DAYS
END
```

The function uses the DEFCENT and YRTHRESH values for the input field HIRE_DATE. In this example, they are set on the field level in the Master File:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,       ALIAS=EID,    FORMAT=A9,                     $
  FIELDNAME=LAST_NAME,    ALIAS=LN,     FORMAT=A15,                    $
  FIELDNAME=FIRST_NAME,   ALIAS=FN,     FORMAT=A10,                    $
  FIELDNAME=HIRE_DATE,    ALIAS=HDT,    FORMAT=I6YMD, DFC=19, YRT=82,  $
.
.
.
```

The function inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

The input values 80 and 81 are less than the threshold 82, so they assume the value 20 for the century. The input value 82 is equal to the threshold, so it defaults to 19 for the century.

The output is:

```
HIRE_DATE     SIXTY_DAYS
80/06/02      2080/08/01
81/07/01      2081/08/30
82/05/01      1982/06/30
82/01/04      1982/03/05
82/08/01      1982/09/30
```

```
82/01/04      1982/03/05
82/07/01      1982/08/30
81/07/01      2081/08/30
82/04/01      1982/05/31
82/02/02      1982/04/03
82/04/01      1982/05/31
81/11/02      2082/01/01
```

Syntax How to Define a Window for a Virtual Field in a Master File

Use standard DEFINE syntax for a Master File, as discussed in your documentation on describing data. Partial DEFINE syntax is shown here.

The parameters DEFCENT and YRTHRESH must immediately follow the field format information.

```
DEFINE fieldname/[format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =
   expression;$
```

where:

fieldname

Is the name of the virtual field.

format

Is a date format such as DMY or YYMD.

DEFCENT

Is the parameter for the default century.

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

YRTHRESH

Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

expression

Is a valid arithmetic or logical expression, function, or function that determines the value of the virtual field.

Example Defining a Window for a Virtual Field in a Master File

In the following example, the DEFINE command in a Master File creates a virtual field named NEW_HIRE_DATE. It is derived from the existing field HIRE_DATE. The format of HIRE_DATE is I6YMD, which is a legacy date with a 2-digit year. NEW_HIRE_DATE is a date format with a 4-digit year (YYMD).

The parameters DEFCENT and YRTHRESH on the DEFINE command create a window from 1982 to 2081, which is used to interpret all 2-digit years for the virtual field. Notice that both DEFCENT and YRTHRESH are coded, as required.

The field-level window takes precedence over any global settings in effect. There is no file-level setting in the Master File.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,       ALIAS=EID,    FORMAT=A9,          $
  FIELDNAME=LAST_NAME,    ALIAS=LN,     FORMAT=A15,         $
  FIELDNAME=FIRST_NAME,   ALIAS=FN,     FORMAT=A10,         $
  FIELDNAME=HIRE_DATE,    ALIAS=HDT,    FORMAT=I6YMD,       $
.
.
.
DEFINE NEW_HIRE_DATE/YYMD DFC 19 YRT 82 = HIRE_DATE;$
```

The following request generates the values in the sample report:

```
TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE
END
```

Since the 2-digit years 80 and 81 are less than the threshold 82, their century assumes the value of DEFCENT + 1 (20), and they are returned as 2080 and 2081 in the NEW_HIRE_DATE column. The 2-digit year 82 is equal to the threshold and therefore defaults to the value of DEFCENT (19). It is returned as 1982.

The output is:

```
HIRE_DATE    NEW_HIRE_DATE
80/06/02     2080/06/02
81/07/01     2081/07/01
82/05/01     1982/05/01
82/01/04     1982/01/04
82/08/01     1982/08/01
```

82/01/04	1982/01/04
82/07/01	1982/07/01
81/07/01	2081/07/01
82/04/01	1982/04/01
82/02/02	1982/02/02
82/04/01	1982/04/01
81/11/02	2081/11/02

Defining a Window for a Calculated Value

Use the DEFCENT and YRTHRESH parameters on a COMPUTE command in a report request to create a window for a temporary field that is calculated from the result of a PRINT, LIST, SUM, or COUNT command. The window is used to interpret a date value for that field when the century is not supplied.

The DEFCENT and YRTHRESH parameters must immediately follow the field format specification; their values are always taken from the left side of the COMPUTE syntax (that is, from the left side of the equal sign). If the expression in the COMPUTE contains a function call, the function uses the DEFCENT and YRTHRESH values for the input field. The standard order of precedence (field level/file level/global level) applies to the DEFCENT and YRTHRESH values for the input field.

In Developer Studio, you can also use the parameters on a COMPUTE command in a MODIFY or Maintain procedure, or on a DECLARE command in Maintain. For details on the use of the parameters in Maintain, see your database maintenance documentation.

Syntax How to Define a Window for a Calculated Value in a Report

Use standard COMPUTE syntax, as described in your documentation on creating reports. Partial COMPUTE syntax is shown here.

On the line that specifies the name of the calculated value, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
TABLE FILE filename
command
[AND] COMPUTE
  fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =
    expression;
.
.
.
END
```

where:

`filename`

Is the name of the file for which you are creating the calculated value.

`command`

Is a command such as PRINT, LIST, SUM, or COUNT.

`fieldname`

Is the name of the calculated value.

`format`

Is a date format such as DMY or YYMD.

`DEFCENT`

Is the parameter for the default century.

`cc`

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

`YRTHRESH`

Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

`yy`

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

`expression`

Is a valid arithmetic or logical expression, function, or function that determines the value of the temporary field.

`END`

Is required to terminate the request.

Syntax How to Define a Window for a Calculated Value in a MODIFY Request

This topic applies to Developer Studio only.

Use standard MODIFY and COMPUTE syntax, as described in your database maintenance documentation; partial syntax is shown here.

On the line that specifies the name of the calculated value, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
MODIFY FILE filename
.
.
.
COMPUTE
  fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =
    expression;
.
.
.
[END]
```

where:

filename

Is the name of the file you are modifying.

fieldname

Is the name of the field being set to the value of *expression*.

format

Is a date format such as MDY or YYMD.

DEFCENT

Is the parameter for the default century.

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

YRTHRESH

Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

expression

Is a valid arithmetic or logical expression, function, or function that determines the value of *fieldname*.

```
END
```

Terminates the request. Do not add this command if the request contains PROMPT statements.

Example

Defining a Window for a Calculated Value

In the following request, the parameters DEFCENT and YRTHRESH on the COMPUTE command define a window from 1999 to 2098. Notice that both DEFCENT and YRTHRESH are coded, as required. The window is applied to the field created by the COMPUTE command, LATEST_DAT_INC.

DAT_INC is formatted as I6YMD, which is a legacy date with a 2-digit year. LATEST_DAT_INC is a date format with a 4-digit year (YYMD). The prefix MAX retrieves the highest value of DAT_INC.

The request is:

```
TABLE FILE EMPLOYEE
SUM SALARY AND COMPUTE
  LATEST_DAT_INC/YYMD DFC 19 YRT 99 = MAX.DAT_INC;
END
```

The highest value of DAT_INC is 82/08/01. Since the year 82 is less than the threshold 99, it assumes the value 20 for the century (DEFCENT + 1).

The output is:

```
SALARY        LATEST_DAT_INC
$332,929.00   2082/08/01
```

Example

Defining a Window for Function Input in a COMPUTE Command

The following sample request illustrates a call to the function JULDAT in a COMPUTE command. JULDAT converts dates from Gregorian format (year/month/day) to Julian format (year/day). For century display, dates in Julian format are 7-digit numbers. The first 4 digits are the century. The last three digits represent the number of days, counting from January 1.

For details on JULDAT, see your documentation on creating reports.

In the request, the input field is HIRE_DATE. The function converts it to Julian format and returns it as JULIAN_DATE. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. JULIAN_DATE is formatted as I7, which is a legacy date with a 4-digit year.

```
TABLE FILE EMPLOYEE
PRINT DEPARTMENT HIRE_DATE
AND COMPUTE
   JULIAN_DATE/I7 = JULDAT(HIRE_DATE, JULIAN_DATE);
BY LAST_NAME BY FIRST_NAME
END
```

The function uses the FDEFCENT and FYRTHRESH values for the input field HIRE_DATE. In this example, they are set on the file level in the Master File:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=82
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,       ALIAS=EID,    FORMAT=A9,          $
  FIELDNAME=LAST_NAME,    ALIAS=LN,     FORMAT=A15,         $
  FIELDNAME=FIRST_NAME,   ALIAS=FN,     FORMAT=A10,         $
  FIELDNAME=HIRE_DATE,    ALIAS=HDT,    FORMAT=I6YMD,       $
.
.
.
```

The function inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

The input values 80 and 81 are less than the threshold 82, so they assume the value 20 for the century. The input value 82 is equal to the threshold, so it defaults to 19 for the century.

The output follows. By default, the second occurrence of the last name SMITH displays as blanks.

```
LAST_NAME    FIRST_NAME   DEPARTMENT   HIRE_DATE   JULIAN_DATE
BANNING      JOHN         PRODUCTION   82/08/01    1982213
BLACKWOOD    ROSEMARIE    MIS          82/04/01    1982091
CROSS        BARBARA      MIS          81/11/02    2081306
GREENSPAN    MARY         MIS          82/04/01    1982091
IRVING       JOAN         PRODUCTION   82/01/04    1982004
JONES        DIANE        MIS          82/05/01    1982121
MCCOY        JOHN         MIS          81/07/01    2081182
MCKNIGHT     ROGER        PRODUCTION   82/02/02    1982033
ROMANS       ANTHONY      PRODUCTION   82/07/01    1982182
SMITH        MARY         MIS          81/07/01    2081182
             RICHARD      PRODUCTION   82/01/04    1982004
STEVENS      ALFRED       PRODUCTION   80/06/02    2080154
```

Additional Support for Cross-Century Dates

The following features apply to the use of dates in your applications.

Default Date Display Format

The default date display format is MM/DD/CCYY, where MM is the month; DD is the day of the month; CC is the first two digits of a 4-digit year, indicating the century; and YY is the last two digits of a 4-digit year.

For example:

```
02/11/1999
```

For a table that fully describes the display of a date based on the specified format and user input, see your documentation on describing data.

Date Display Options

The following date display options are available:

- You can display a row of data, even though it contains an invalid date field, using the command SET ALLOWCVTERR. The invalid date field is returned as the base date or as blanks, depending on other settings. For details, see your documentation on the SET command. This feature applies to non-FOCUS data sources when converting from the way data is stored (ACTUAL attribute) to the way it is formatted (FORMAT or USAGE attribute).
- If a date format field contains the value zero (0), you can display its base date, using the command SET DATEDISPLAY = ON. By default, the value zero in a date format field such as YYMD is returned as a blank. For details, see your documentation on the SET command.
- You can display the current date with a 4-digit year using the Dialogue Manager system variables &YYMD, &MDYY, and &DMYY. The system variable &DATE*fmt* displays the current date as specified by the value of *fmt*, which is a combination of allowable date options, including a 4-digit year (for example, &DATEYYMD). For details, see your documentation on Dialogue Manager.

System Date Masking

You can temporarily alter the system date for application testing and debugging, using the command SET TESTDATE. With this feature, you can simulate clock settings beyond the year 1999 to determine the way your program will behave. For details, see your documentation on the SET command.

Date Functions

The date functions supplied with your software work across centuries. Many of them facilitate date manipulation. For details on date functions, see *Using Functions*.

Date Conversion

You can convert a legacy date to a date format in a FOCUS data source using the option DATE NEW on the REBUILD command. For details, see your documentation on database maintenance.

Century and Threshold Information

The ALL option, in conjunction with the HOLD option, on the CHECK FILE command includes file-level and field-level default century and year thresholds as specified in a Master File. For details, see your documentation on describing data.

Date Time Stamp

The year in the time stamp for a FOCUS data source is physically written to page one of the file in the format CCYY.

Index